Amortized analysis is a technique used in computer science to average the time complexity of an algorithm over a sequence of operations, rather than analyzing the cost of each individual operation separately. This approach provides a more accurate assessment of an algorithm's performance in practice, especially when the cost of operations varies significantly.

Key Concepts

1. **Amortized Cost**: This is the average cost per operation over a sequence of operations. It helps in smoothing out the high costs of some operations by spreading them across many low-cost operations.

2. **Aggregate Method**: In this method, the total cost of a sequence of operations is computed and then divided by the number of operations to obtain the average cost.

3. **Accounting Method**: This method assigns different costs to different operations, sometimes charging more for an operation than its actual cost. The extra charge is stored as "credit" that can be used to pay for more expensive operations later.

4. **Potential Method**: This method involves defining a potential function that maps the state of a data structure to a real number. The amortized cost is then the actual cost plus the change in potential. This method is particularly useful for dynamic data structures where the state changes over time.

Example

Consider a dynamic array (like a Python list or Java ArrayList) that doubles its size when it runs out of space:

- **Cost of inserting an element**: Normally, inserting an element takes constant time, $O(1)$. However, when the array is full, a resize operation is needed, which involves creating a new array and copying all elements to it. This resize operation is costly, taking $O(n)$ time for n elements.

- **Amortized Analysis**: Over a sequence of insertions, the amortized cost per insertion is still $O(1)$ because the costly resize operation happens infrequently, and its cost is distributed across all subsequent insertions until the next resize.

Advantages

- **Realistic Performance Measurement**: Amortized analysis provides a more realistic measure of an algorithm's performance by considering sequences of operations.
- **Better Worst-Case Guarantees**: It can offer better guarantees than analyzing the worst-case cost of a single operation.

Amortized analysis is widely used in data structures like hash tables, dynamic arrays, and various tree structures to ensure efficient average performance.